OULU UNIVERSITY OF APPLIED SCIENCES

Erik Maresia

TIME SERIES FORECASTING WITH LONG SHORT-TERM MEMORY NEURAL NETWORKS ON THE STOCK MARKET

TIME SERIES FORECASTING WITH LONG SHORT-TERM MEMORY NEURAL NETWORKS ON THE STOCK MARKET

Erik Maresia
Bachelor's Thesis
Spring 2020
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Information Technology

Author: Erik Maresia
Title of the bachelor's thesis: Time Series Forecasting with Long Short-Term Memory Neural Networks on the Stock Market
Supervisor: Pekka Alaluukas
Term and year of completion: Spring 2020 Number of pages: 40

The stock market is notoriously difficult to predict, but there are two schools of thought that make approximation possible. Both the fundamental and technical analysis attempt to provide ways in which a human or a machine can predict the market.

Machine learning is an efficient tool to use for this problem. It offers solutions that fit the scope of the problem, such as long short-term memory neural networks. Combining all the knowledge of economics and machine learning, it is possible to attempt to overcome the efficient-market hypothesis, which states that it is impossible to outperform the market.

The aim of this bachelor's thesis is to demonstrate how a neural network works and how it may be used to solve a problem such as stock market forecasting. Since it is not known how neural networks find patterns, this thesis will give a theory for stock markets to demonstrate what might be happening inside a neural network when it finds patterns.

Keywords: Machine Learning, Neural Networks, Time-Series Forecasting, Efficient Market Hypothesis

PREFACE

I would like to thank all those who supported me in my thesis, especially my supervisor Pekka Alaluukas and my language teacher Kaija Posio for their continuous guidance and support.

Oulu, 28.1.2020
Erik Maresia

CONTENTS

ABBREVIATIONS

EMH	Efficient Market Hypothesis
GGM	Gordon Growth Model
LSTM	Long Short-Term Memory
MSE	Mean Squared Error
NASDAQ	National Association of Securities Dealers Automated Quote System
P/E Ratio	Price-to-Earnings Ratio
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SMA	Simple Moving Average

1 INTRODUCTION

Machine learning is changing the world. It is a method of teaching computers to find patterns and to predict future patterns using a given data set. There are two main methods for a machine to learn: supervised and unsupervised. Supervised learning is done with a data set which is labelled. This means that for a given input, the computer knows what the output should be. Examples of supervised learning are predicting the weather forecast, the stock market and housing prices. Unsupervised learning is done by finding unknown structures, groups and patterns in the input data by finding similarities in it. The input data is not labelled, therefore the machine has to find the groups and structures itself. An example of unsupervised learning is grouping customers by a purchasing behaviour. (Brownlee 2016b.)

The aim of this thesis is to lay the foundation to understand the architecture of a neural network and to forecast Apple Inc.'s stock market close prices. This thesis uses the word time-series to describe successive data points ordered chronologically and will use Apple Inc. stock data as time-series data. In the following chapters, an introduction is set for plain neural networks and recurrent neural networks, respectively.

While it is known what a neuron in a given layer is doing, it is not known how a network finds the patterns it finds. Chapter 2 will lay out the theory on stock markets and how stock prices may be predicted mathematically to demonstrate possibilities what might be happening inside a neural network when it is learning to find patterns.

To fully understand this thesis a knowledge of mathematics is required – especially in the areas of linear algebra and calculus. While all mathematical formulae are explained, knowledge in these mathematical areas help to understand the mathematical symbols used and how certain formulae work.

2 STOCK MARKET

Publicly held companies may raise capital by selling a portion, or a share, of their company to prospective investors. These shares entitle new shareholders to profits made by the company. Profits generally take the form of appreciation of value or are distributed in cash. Distributed profits are called dividends. (Chen 2018.)

The stock market is a collection of markets and exchanges where buying and selling of publicly held companies happens (Chen 2019). The shares of a publicly held company are traded in a stock exchange. There are many stock exchanges in the world, most notably the New York Stock Exchange, the Tokyo Stock Exchange, the London Stock Exchange and the NASDAQ. While all stock exchanges have the common interest of buying and selling shares, they differ in their organizational structure and in the way shares are traded. (Pilbeam 2010, 206.)

2.1 Efficient Market Hypothesis

Although stock prices are essentially driven by supply and demand, the efficient market hypothesis states that the price of a stock reflects all available information (Mankiw & Taylor 2014, 548). This means that any change in the affecting variable of a stock will have an immediate effect on the price of the stock. In practice the hypothesis means that it is impossible to predict the stock market because it is not possible to have all the available information present (Pilbeam 2010, 239). Predictable changes in stock prices are therefore nearly non-existent (Mishkin 2018, 163).

2.2 Stock Prediction Methodologies

As the efficient market hypothesis states, forecasting the price of a share accurately is impossible. What is possible, however, is to obtain a relatively close prediction. These predictions are used constantly by financial institutions worldwide.

There are two main methodologies in stock market forecasting on which investors and financial institutions rely: fundamental analysis and technical analysis. The efficacy of both analyses is disputed by the efficient market hypothesis mentioned earlier, but they can give a close prediction (Majaski 2019).

2.2.1 Fundamental Analysis

The fundamental analysis uses the financial statements of a company and a stock to analyse and forecast the price of a stock's price. Anything that can affect the value of the stock is studied (Mankiw & Taylor, 548). There are many formulae which are used in the fundamental analysis of a company, but this thesis will explore only two.

Gordon Growth Model

The Gordon Growth model (GGM) is a method of fundamental analysis for calculating whether a stock is overvalued or undervalued. GGM assumes that the growth rate of a stock's dividend is constant. (Pilbeam 2010, 218.)

FORMULA 1. Gordon Growth Model equation

$$P = \frac{D_1}{r - g}$$

where

P = intrinsic value of stock

D_1 = next year's expected dividend

r = rate of return

g = expected constant growth rate

By using the formula shown in Formula 1, it is possible to evaluate whether a stock is under- or overpriced by comparing the answer to the actual price of the stock.

P/E Ratio

The price-to-earnings ratio (P/E ratio) is used for valuing a company that measures its current stock price relative to its per-stock earnings. It is calculated by dividing a company's market value per share by the earnings per share, as shown in Formula 2. The higher the P/E ratio is, the more the market values the company, and vice versa. On the other hand, a low P/E ratio could mean in practice, for instance, that a company's stock is expected to have poor future earnings performance (Pilbeam 2010, 230).

FORMULA 2. Price-to-earnings formula

$$P/E\ Ratio = \frac{Market\ value\ per\ share}{Earnings\ per\ share}$$

It is important to note that if a company is losing money, there is no P/E ratio, since there are no earnings per share.

2.2.2 Technical Analysis

The technical analysis only makes use of historical data of the stock to forecast the direction of the price. The analysis of a company's stock from this point-of-view disregards fundamental factors (Pilbeam 2010, 240). A close view of a company's financial statement is not necessary. Because it is already reflected in the price, there is no need to inspect all factors separately. The only thing that remains is the analysis of price movements. This is the product of supply and demand. (Hayes 2019.)

The technical analysis is based on the search and careful analysis of trends and patterns. Only a few common trends and patterns are discussed and explained in this thesis.

Momentum

The momentum of a share's price is the velocity at which the price changes. It is measured by constantly taking the price differences for a fixed time period as presented in Formula 3. For example, for a 10-day momentum line, it is necessary to take the closing price from 10 days before the calculation is made and subtract it from the latest closing price. (Murphy 1999, 299.)

FORMULA 3. Equation for a share's momentum

$$M = V - V_x$$

where
M
$M = V - V_x$
M = the momentum of a share's price
V = latest closing price of a share
V_x = closing price of a share x days ago

Simple Moving Average

The simple moving average (SMA) is a method used to show the real trend of the stock and disregard sudden price fluctuations (Pines 2019). The formula for calculating the SMA is shown in Formula 4.

FORMULA 4. Simple moving average

$$SMA = \frac{1}{n} \sum_{i=0}^{n} A_i$$

where

n = number of time stamps collected for SMA

i = current closing price

A = share

The more time stamps are taken into the calculation, the less the average will be affected by sudden price fluctuations, and vice versa (Figure 1).

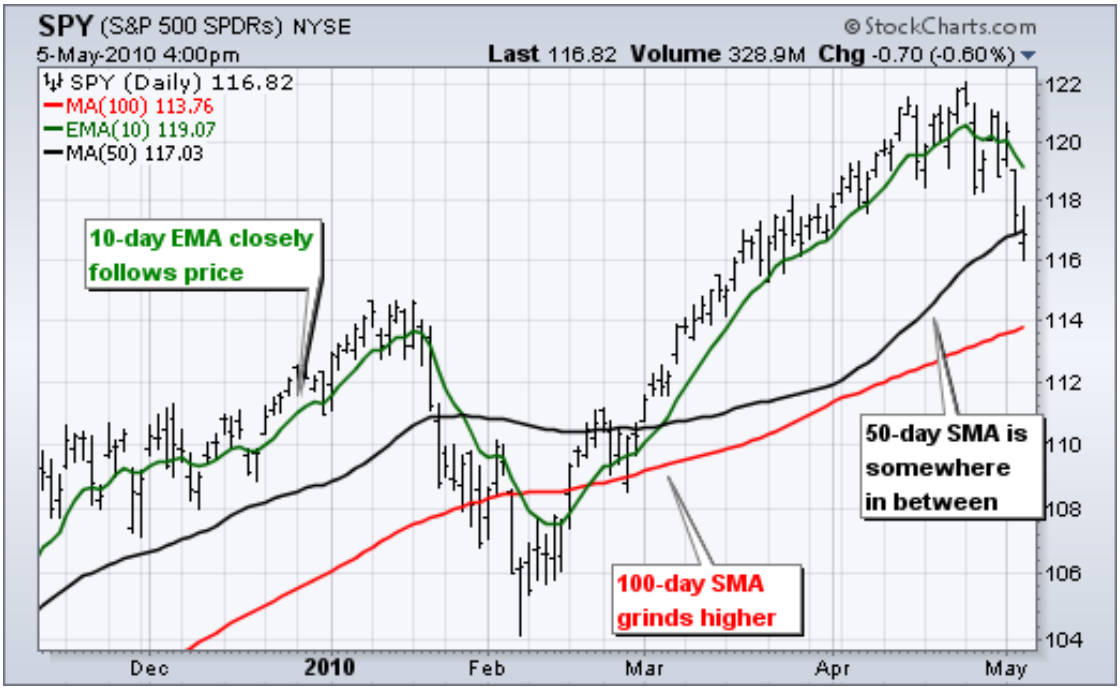

FIGURE 1. SMA of S&P index with different time stamps (StockCharts 2020)

3 NEURAL NETWORKS

Neural networks are an example of machine learning, where a program can change while it learns a problem. These problems may include image recognition and predicting housing prices (supervised learning), or customer segmentation (unsupervised learning). (DeMuro 2019.) This chapter will explain a so-called plain, or a feedforward neural network, which will use labelled data as its input (supervised learning).

Modelled after the human brain, neural networks are computing devices that are interconnected by many small processors which all have a small simple task. Each small processor is only aware of the information it receives and of the information it sends forward. These processors are called neurons and they are linked together by so-called weighted connections. (Callan 1999, 1.) What happens between the input and output of a neural network is a *black box* – it is not known exactly why a neural network outputs what it does (Heaton 2015, 35).

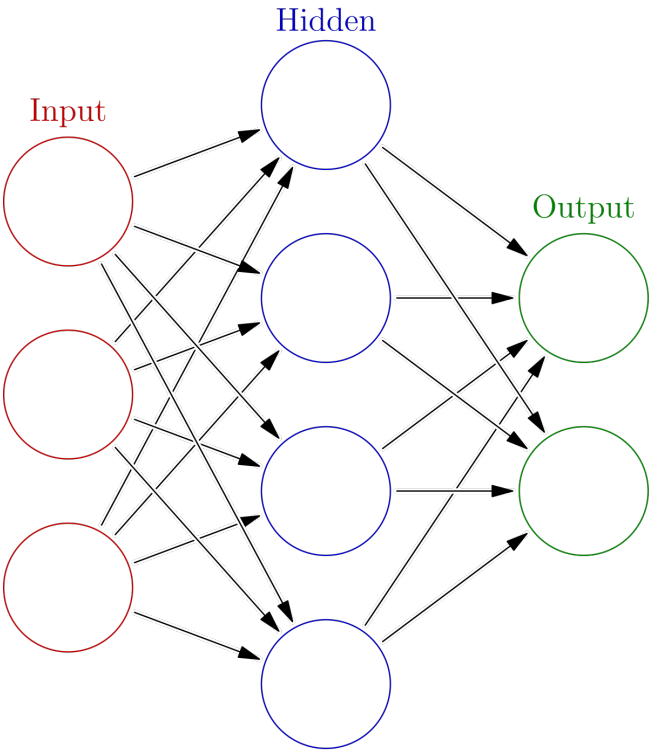

FIGURE 2. Architecture of a neural network (Wikipedia 2020)

3.1 Neuron

A neuron is the individual interconnected unit that forms most neural networks. It will receive an input as an integer or a floating-point number. There are different kinds of neurons in a neural network (Figure 2). The input neurons pass information into the neural network, while the output neurons pass information out of the neural network. The hidden neurons, which are between the input and output neurons, help the neural network process information in various ways. (Heaton 2015, 25.)

Each neuron has its individual internal state: activation and activity level. This state works as a function of the number of inputs it receives. (Haykin 1994, 9.) A single neuron, shown as a ball in the centre of Figure 3, will receive an *N* number of inputs from the previous layer: x_1, x_2 until x_N from their respective neurons. The connections from x_1, x_2 until x_N all have weights (*w*), which are real numbers – measurements to show the importance of the respective input. The neuron will receive its output *y* from its activation function. (Nielsen 2019.)

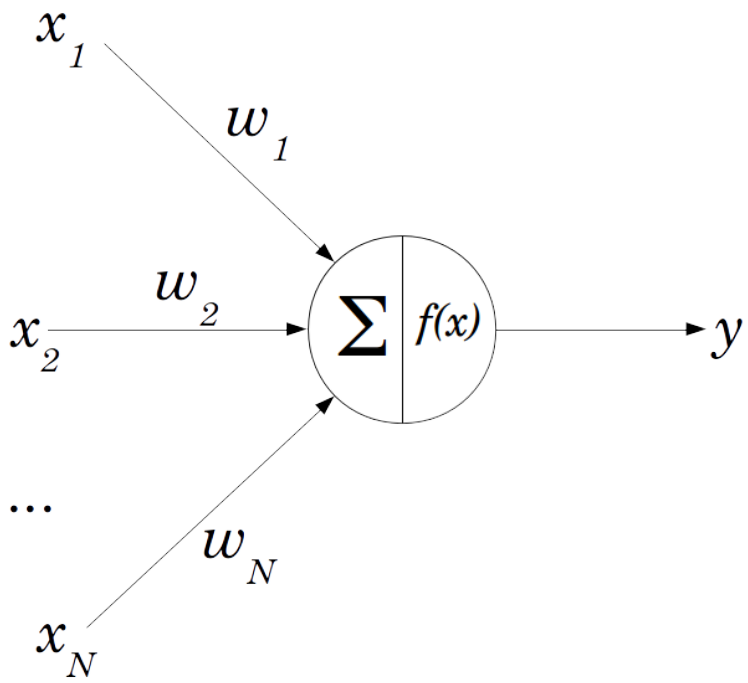

FIGURE 3. Diagram of a simple artificial neuron (Lucidarme 2018)

Sometimes it is desirable to activate a certain neuron with more ease than others. This can be achieved with a value called the *bias*. This bias is added to the summation of the inputs and weights of the neuron. (Nielsen 2019.) A summation, or the linear combination of all the inputs, weights and the bias, is done in the neuron before calling the activation function (Formula 5).

FORMULA 5. A neuron's input summation

$$u_k = \sum_{j=0}^{p} w_{kj} x_j + b$$

where

k = neuron in question

u = linear combining output

p = number of inputs

j = index of input vector

w = synaptic weights as a matrix

x = input signal as a vector

b = bias

3.2 Activation Functions

The activation function inside an artificial neuron is the function which will take as an input the weighted sum of the inputs and the bias of the neuron (Formula 5) and convert them into the neuron's output value (Freeman & Skapura 1992, 19). There are a number of different activation functions, but this thesis will discuss only three: sigmoid, hyperbolic tangent and rectified linear unit.

3.2.1 Sigmoid

The sigmoid function is a very common activation function used in neural networks for cases where the output of the neuron is desired to be a positive number (Heaton 2015, 14). The output of a sigmoid function will always be between 0 and 1 (Formula 6) (Figure 4).

FORMULA 6. The sigmoid function as an equation

$$\sigma(x) = \frac{1}{1 + e^x}$$

where

x = input to a neuron

e = natural logarithm base (Euler's number)

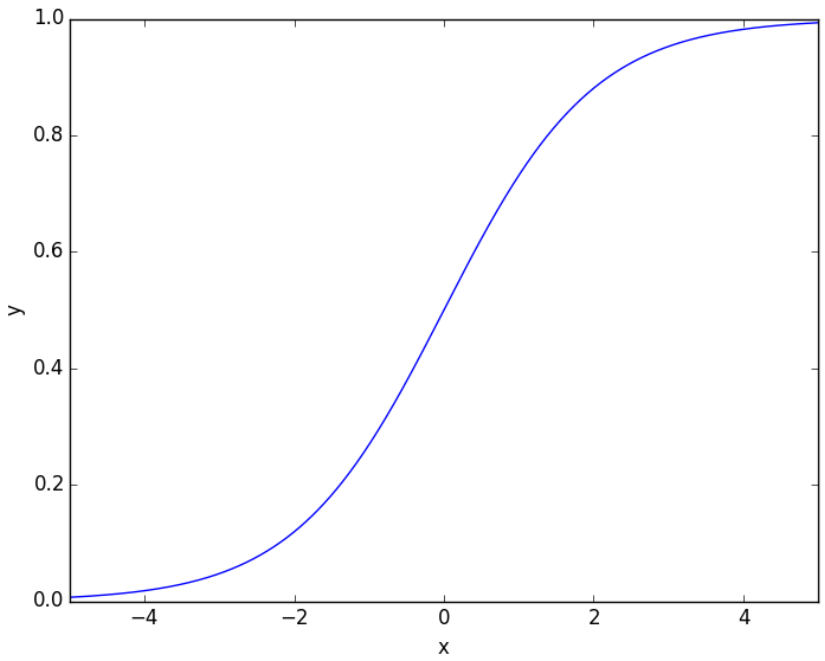

FIGURE 4. The sigmoid function (Brixius 2016)

3.2.2 Hyperbolic Tangent

While the hyperbolic tangent has a sigmoid-like shape (Figure 5), it differs from the sigmoid function discussed above in that it will output a value between -1 and 1. Strongly positive numbers are therefore mapped as a positive output and vice versa. (De Marchi & Mitchell 2019, 51.) The hyperbolic tangent can be calculated using Formula 7.

FORMULA 7. Hyperbolic tangent equation

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

where

x = input to a neuron

e = natural logarithm (Euler's number)

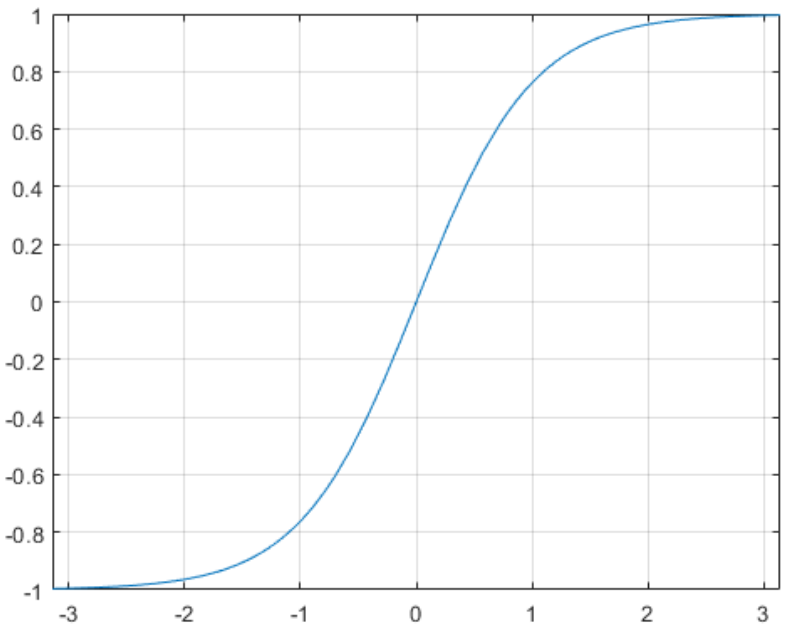

FIGURE 5. Hyperbolic tangent line (MathWorks 2019)

3.2.3 Rectified Linear Unit

The sigmoid and hyperbolic tangent activation functions both share the same problem of saturation. This means that very large values will output 1, while very small values will output 0 for sigmoid and -1 for the hyperbolic tangent. The Rectified Linear Unit (ReLU) fixes this issue. If the value given to it is less than or equal to 0, it will output 0, otherwise it will output the value it was given (Formula 8). In this way, the ReLU can be a linear activation (Figure 6) function which has advantages over the nonlinear proponents mentioned earlier. (Brownlee 2019a.)

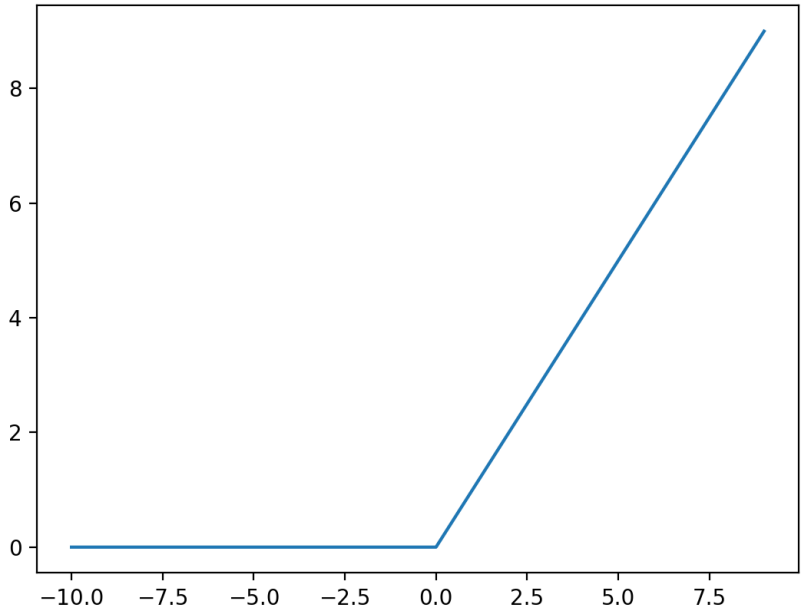

FIGURE 6. ReLU line (Brownlee 2019a)

FORMULA 8. Rectified Linear Unit equation

$$f(x) = max\ (0, x)$$

where

x = the input to a neuron

3.3 A Neural Network Learning

The primary role of a neural network is to learn from its environment and to be able to improve its performance by learning (Haykin 1994, 45). It learns by essentially finding all the right weights and biases (Abdi, Valentin & Edelman 1999, 2). This is a long iterative process, involving many different steps, essentially done by minimizing the cost function of the neural network (Haykin 2019, 47).

3.3.1 Cost Function

The cost function, or loss function, evaluates how accurate or inaccurate the neural network is (Nielsen 2019). There are many different cost functions, which attain the same objective, but this thesis will only consider the mean squared error (MSE) cost function. The cost function will return a value that is called an error, which shows how wrong the neural network is in predicting the wanted outcome (Haykin 1994, 47). It is computed by taking the average of the squared difference of the vector from each output neuron by the expected output vector (Formula 9). The higher the error is, the lower the probability is that the neural network is right in its predictions.

FORMULA 9. Mean squared error

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2$$

where

N = number of predictions

i = index in a vector

x = predicted values as a vector

y = expected values as a vector

3.3.2 Gradient Descent

The error returned from the cost function can be minimized using an optimizer algorithm known as gradient descent (Brownlee 2016a). An example of a cost function *C* can be seen in Figure 7 with two variables: v_1 and v_2. The gradient descent will try to find the values for v_1 and v_2 that will minimize the value of the function *C*. It is important to note that the cost function *C* shown in Figure 7 is a 3-dimensional function. Many real-world neural networks may have several thousands to millions of dimensions, depending on the architecture of the neural network. For the sake of clarity and visualization, a 3-dimensional example is used in this chapter.

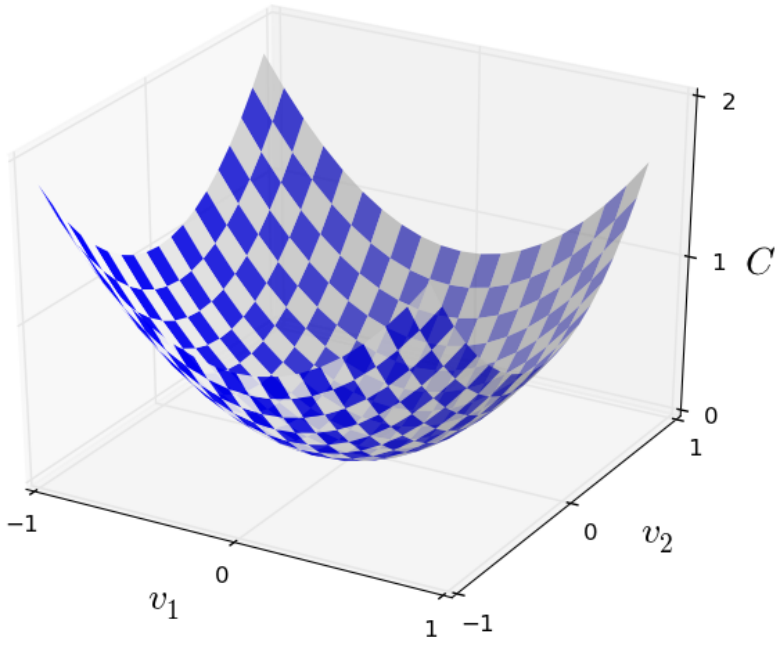

FIGURE 7. Function C*, with variables* v1 *and* v2 *(Nielsen 2019)*

Using the equation presented in Formula 10, it is possible to find the direction of the steepest ascent by taking the gradient vector of the function seen in Figure 7 (Nielsen 2019). Another way of interpreting this would be by asking the question: What direction, at any given point, is the fastest way up?

FORMULA 10. Gradient of a function C

$$\nabla C = \left(\frac{\partial C}{\partial v_1} \quad \frac{\partial C}{\partial v_2}\right)^T$$

where

v = variable of function C

T = the transpose of a 2-dimensional vector to a 1 x 2 -dimensional matrix

The algorithm of gradient descent, however, is not interested in the steepest ascent (Formula 10), which would maximize the error, but the steepest *descent* – the minimization of the error. Therefore, the negative gradient is used.

3.3.3 Backpropagation

Backpropagation is the core algorithm to how neural networks learn. Essentially, backpropagation will use gradient descent to change the weights and biases of the network in order to minimize the error of the cost function (Haykin 1994, 66). It will traverse through the network starting from the output layer, proceeding all the way back to the input layer. This process is done iteratively until the gradient of the cost function converges to a local or global minimum, a minimum being either the minimum value of the function or the minimum value of the function in a given range. (Bishop 1996, 140-141; Brownlee 2016a.)

While the activations of neurons cannot be directly changed, since they are computed values (Chapter 3.2), the weights and biases leading to those activations can be. Formula 11 shows, how the partial derivative of C_0 changes with respect to the change in the activation of the node in the previous layer. In other words, Formula 11 shows the sensitivity of C_0 to changes in $a_k^{(L-1)}$.

FORMULA 11. Backpropagation equation

$$\frac{\partial C_0}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_{L-1}} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}}$$

where

C_0 = cost function of a single input

a = activation of a neuron (Chapter 3.2)

L = the last layer of neurons

k = index of a neuron in the $L - 1$ layer

n = number of neurons in a layer

z = weighed sum of a neuron (Formula 7)

j = index of a neuron in the L layer

4 RECURRENT NEURAL NETWORKS

While a plain neural network (Chapter 3), also known as a feedforward neural network, is the basis of all subsets of neural networks and is useful in many cases of machine learning, it fails in forecasting sequential data. This is because a plain neural network is aware only of the information that is present in that input of the network. Recurrent neural networks (RNNs) were invented to solve this problem and are used in a number of different machine learning tasks, such as predicting a word in a sentence, time-series forecasting and speech recognition – all being cases where plain feedforward neural networks cannot succeed as well. (Banerjee 2018.)

RNNs use a short-term memory-based architecture, where outputs of a neuron are not only sent to the output layer of the network but are also fed back to the hidden layer into the next time-step (Figure 8) (Kostadinov 2018, 7-8). In this way, the network will process each input in the context of previous inputs (Kelleher 2019, 171). As an example, an RNN will analyze a sentence sequentially word by word. In order for the network to understand the context of a sentence, it must feed the previous word back into the network to understand the next word.

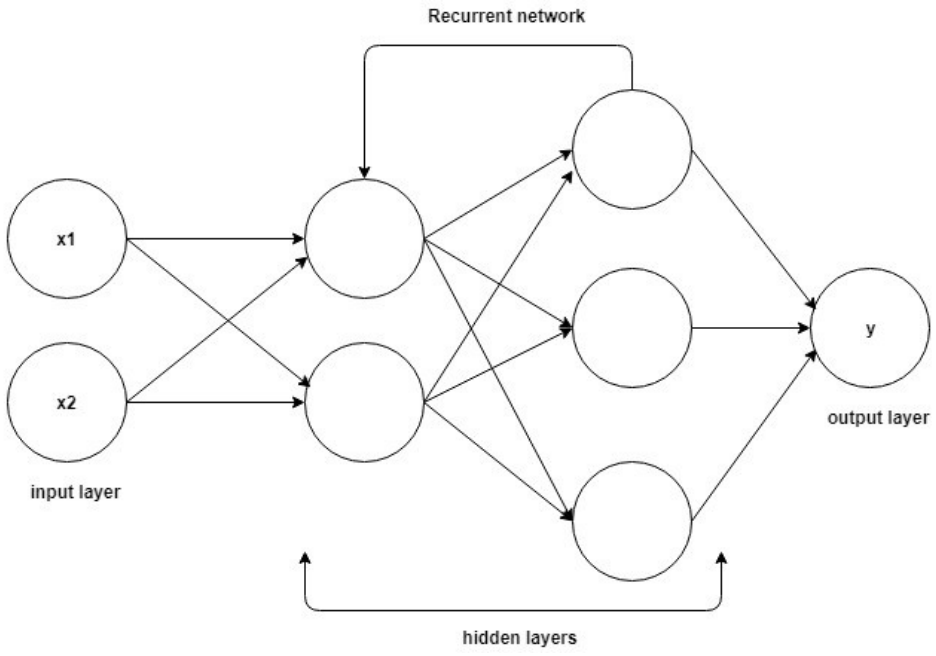

FIGURE 8. Architecture of a recurrent neural network (Debarko 2018)

4.1 Vanishing Gradient Problem

As a neural network is learning, it updates its weights and biases to minimize the gradient of the cost function by backpropagating through the network (Chapter 3.3). Neural networks with many layers will have difficulty in maintaining a gradient that has an effect (Brownlee 2019b). When calculating the gradient with respect to the activation function (Formula 11), the gradient tends to become so small that by the time the network has propagated back to the first layer, it will have little to no effect on setting new weights and biases. A vanishing gradient does not allow the neural network to learn as it should. (Ravichandiran 2018, 152.)

4.2 Long Short-Term Memory

Long short-term memory (LSTM) was invented as the answer to the problem of the vanishing gradient problem. It is a type of recurrent neural network which can learn with long-term dependencies without a vanishing gradient. Just as regular RNNs have a chain-like structure of passing information back into the layer (Figure 8), LSTMs are no different, except for the structures through which the information is passed (Figure 9). (Olah 2015.) In the following subchapter the subscript t is denoted as the index of the time-step of the LSTM network. This thesis will focus on long short-term memory neural networks.

4.2.1 An LSTM Cell

An LSTM network consists of individual LSTM cells (Figure 9), all with an individual state. This state plays a crucial role in handling long-term dependencies. The cell receives three inputs: the state of the cell from the previous time-step (C_{t-1}), the output of the cell in the previous time-step (h_{t-1}), and the input from the previous layer (x_t). These inputs pass through different gates and functions within the cell to determine the outputs and the state of the cell. There are three different gates in an LSTM cell. They determine what information is forgotten from the state of the cell, what is updated to the state of the cell, and what the output of the LSTM cell is. (Ravichandiran 2018, 153.)

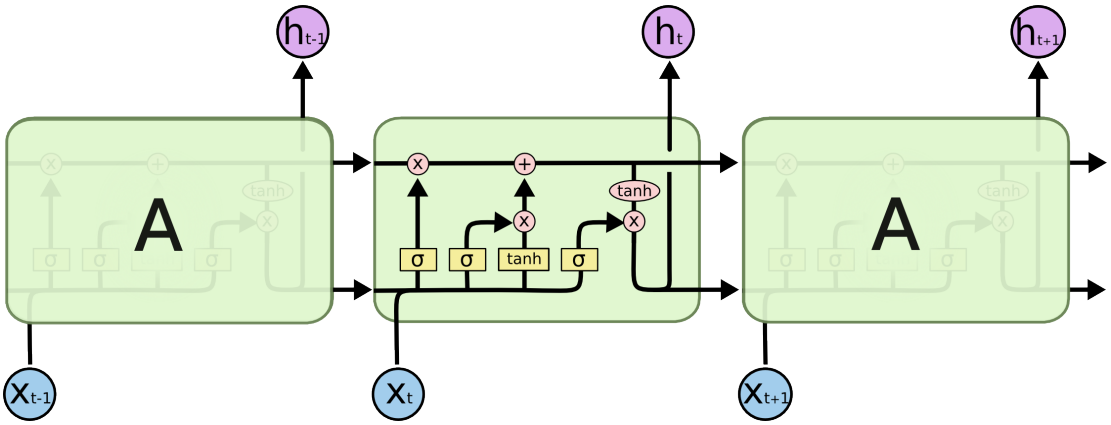

FIGURE 9. LSTM layer with three LSTM cells at three different time-steps t *(Olah 2015)*

Forget Gate

When a piece of information is passed from the previous cell, the LSTM cell will decide what to do with that piece of information (Figure 10) (Ravichandiran, 153). The input from the previous layer x_t and the input from the cell in the previous time-step h_{t-1} are concatenated into a vector. This vector will then go through the forget gate f_t – a sigmoid function (Chapter 3.2.1) – which is responsible for deciding what information should not be added to the cell state. The forget gate will output a new vector of numbers ranging between 0 and 1, which represents the importance of the information in the cell state of the previous time-step, 0 being expendable and 1 being important. This new vector will be concatenated with the cell state of the previous time-step C_{t-1} by an element-wise product, which will decrease values of lesser importance, and vice versa. (Olah 2015.)

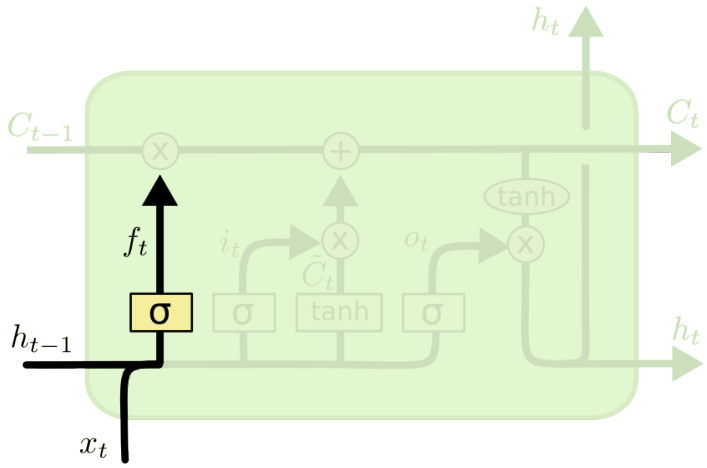

FIGURE 10. Forget gate of an LSTM cell, where t *depicts the time-step (Olah 2015)*

Input Gate

After the forget gate has updated the cell state, the LSTM cell will decide what values to update the cell state with by using an input gate (Ravichandiran, 154). There are three steps at which the input gate decides what values will be updated to the state (Figure 11). Firstly, the concatenated vector of h_{t-1} and x_t will go into a sigmoid function to receive a vector of values for the cell to update (i_t). This vector will have values between 0 and 1 to show the importance of a value in the potential cell state ($\tilde{C}_t$). Secondly, the concatenated vector goes through a hyperbolic tangent function (Chapter 3.2.2) to receive a potential cell state ($\tilde{C}_t$). These two vectors are then concatenated by an element-wise product. Lastly, the new concatenated vector of $\tilde{C}_t$ and i_t is added to the cell state. The new cell state can therefore be expressed with Formula 12. (Olah 2015.)

FORMULA 12. Update equation for an LSTM cell

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

where

C = cell state as a vector

f = vector of values to forget from the forget gate

i = vector to decide what values to update

$\tilde{C}$ = potential cell state as a vector

t = current time-step

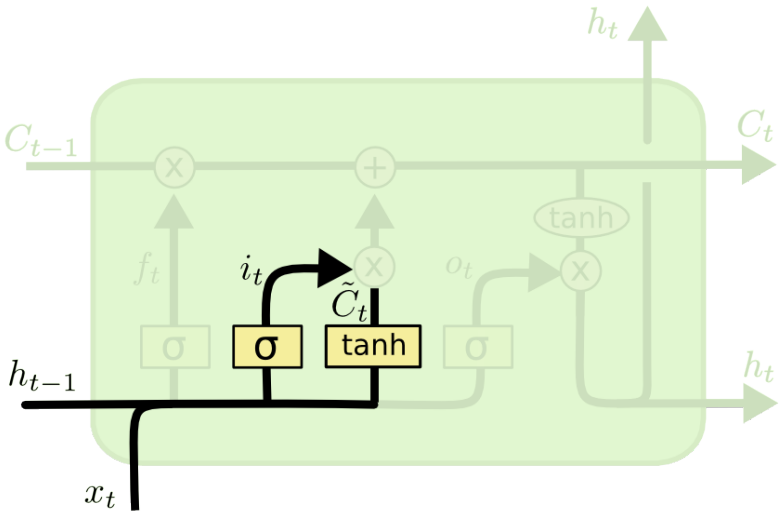

FIGURE 11. Input gate of an LSTM cell (Olah 2015)

Output Gate

The output of the LSTM cell is derived from two steps (Figure 12). Firstly, the concatenated vectors of h_{t-1} and x_t will go through a sigmoid function to determine what values of the cell state are important. Secondly, the cell state will go through a hyperbolic tangent function to receive the vector in the range between -1 and 1. These two vectors are then concatenated by an element-wise product to form a new vector h_t (Formula 13). This new vector will then continue to the next layer of the network as the output and to the next LSTM cell at time-step *t +1*. (Olah 2015; Kang 2017.)

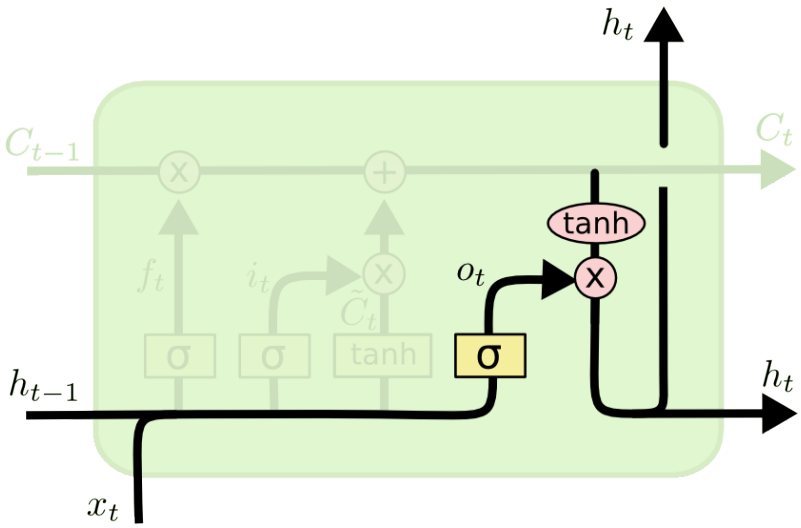

FIGURE 12. Output gate of an LSTM cell (Olah 2015)

FORMULA 13. An LSTM cell's output equation

$$h_t = o_t \odot tanh(C_t)$$

where

h = output vector

o = sigmoid function of vectors of h_{t-1} and x_t

tanh = hyperbolic tangent function

C = cell state

t = time-step

5 FORECASTING MARKET CLOSE PRICES WITH LSTM

This chapter will aim to demonstrate how LSTM neural networks can be used to conduct time-series forecasting, such as stock prices. The previous chapters in this thesis have explained the theory behind both the stock market and neural networks. Since it is not known how neural networks find the patterns they find, Chapter 2 gave an introduction to how it is possible to forecast stock prices mathematically. These formulae presented in Chapter 2 help understand what might be happening in the hidden layers of a neural network.

5.1 Tools and Libraries

Machine learning applications are developed mostly in Python. Python is a high-level general-purpose programming language, which is optimal for machine learning since it is easy to understand. Machine learning can be developed in any programming language, but Python has gained its dominance in the field because of its simple syntax. (Protasiewicz 2018.)

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from keras.models import Sequential
5 from keras.layers import LSTM, Dense
6 from sklearn.preprocessing import MinMaxScaler
```

FIGURE 13. External libraries used

Neural network algorithms may be constructed manually, but since there is a high demand for machine learning, software libraries have been developed to do the heavy lifting. Keras is an open-source neural network library which is written in Python and will be used in this thesis. Keras in itself offers many packages, such as models and layers (Figure 13, lines 4 and 5). The Keras packages used in this thesis will be explained further on.

There are several libraries for data manipulation, analysis and visualization in Python (all seen in Figure 13). NumPy was created as a linear algebra library and will be used for all matrix and vector operations in this thesis. The data analysis

will be done with Pandas, which is a software library for Python specialized for data manipulation of time-series. Matplotlib will be used for data visualization.

To make a neural network learn faster and to ensure that one feature does not rule over another, data must be normalized (Boris 2019). This is done by rendering all the values of the data between 0 and 1. MinMaxScaler from SkLearn is a utility library for this task as it will input any data and normalize it.

5.2 Data

The data used for predicting stock closing prices is historical stock data from Apple Inc. in the format seen in Table 1. Stock data of a specific stock is represented with several indicators: the opening and closing price of the stock for that day, the lowest and highest prices for that day and the volume in which the stock was traded in that day. These indicators are called features in a neural network's input. This thesis uses Apple Inc's stock data from 06.10.2011 until 18.09.2019.

TABLE 1. Portion of the data set of Apple Inc. stock history

Date	Open ($)	High ($)	Low ($)	Close ($)	Volume
11.11.2013	74.284286	74.524284	73.487144	74.150002	56863100
12.11.2013	73.952858	74.845711	73.857140	74.287140	51069200
13.11.2013	74.000000	74.607140	73.851425	74.375717	49305200
14.11.2013	74.687141	75.611427	74.552856	75.451431	70604800
15.11.2013	75.225716	75.584282	74.927139	74.998573	79480100

This thesis will only be concerned with forecasting the close prices of Apple Inc.'s stock. A new column will be added to Table 1, which will have the value of the next day's close price. By labelling the data in this way, the neural network may learn to see patterns about what features may have an effect on the closing price. The predictive model created in this thesis will only use the previous day's stock information to predict the next.

The data will be split into two, training and testing data, where training data will be 60% of the total data. The neural network will first learn with the training data

and when the model is accurate enough, it will try to predict new values using the testing data.

5.3 Constructing a Neural Network

To initialize a neural network, the Sequential method must be called from Keras' models library (Figure 13 line 4 and Figure 14 line 1) which will initialize a new neural network. An LSTM layer is added to this neural network (Figure 14 line 2) together with an additional Dense layer (Figure 14 line 3), which is a plain neural network layer. The model is then compiled using the mean squared error as the loss function and gradient descent as the optimizer algorithm used for minimizing the error of the loss function.

```
1 model = Sequential()
2 model.add(LSTM(units=60, input_shape=(1, 5)))
3 model.add(Dense(1))
4 model.compile(loss='mse', optimizer='sgd')
```

FIGURE 14. Creating the neural network

For the model to learn, it is then run through a number of epochs. An epoch is an iteration over the dataset which makes the neural network learn. This thesis will show the difference between two different epoch values 10 and 100. With 10 epochs, the error of the loss function declines but does not reach the minimum, which is 0 (Figure 15).

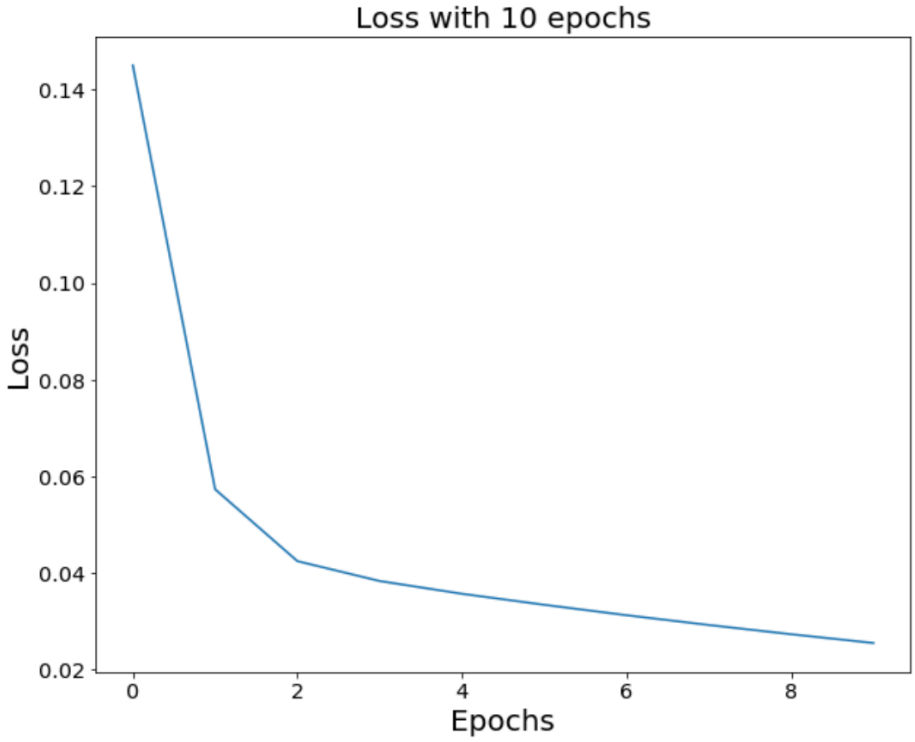

FIGURE 15. Loss of the model with 10 epochs

The number of epochs the model iterates over the data may have effects on the predictions. As mentioned in Chapter 3.3.3, a neural network will be most effective when the error is at the lowest value possible. With 10 epochs, the predictions of the neural network are not accurate, with only small trends seen between the predicted values and the actual values (Figure 16).

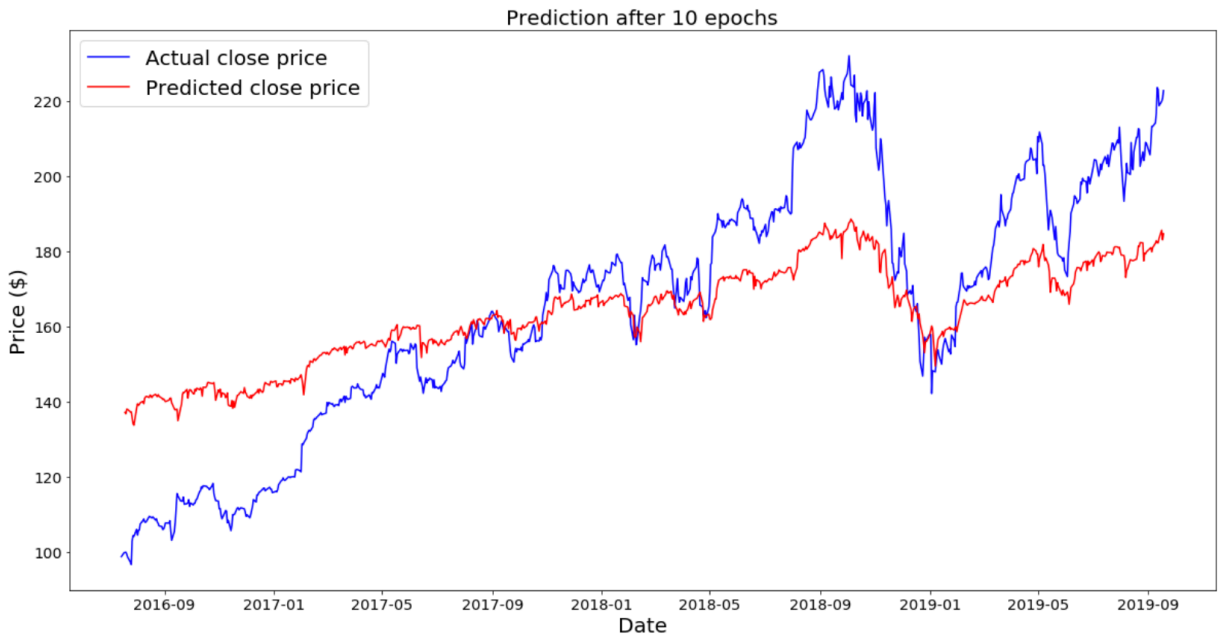

FIGURE 16. Prediction of the model with 10 epochs

When the epoch count is increased to 100, the loss function reaches its minimum of 0 (Figure 17). This increase in epochs and the minimization of the loss function are shown clearly in the prediction of the neural network (Figure 18). While the predictions are not entirely correct in every time-step, the neural network correctly predicts trends. The predictive model, which is demonstrated by this thesis only, uses the previous day's stock information to predict the next day's market close price.

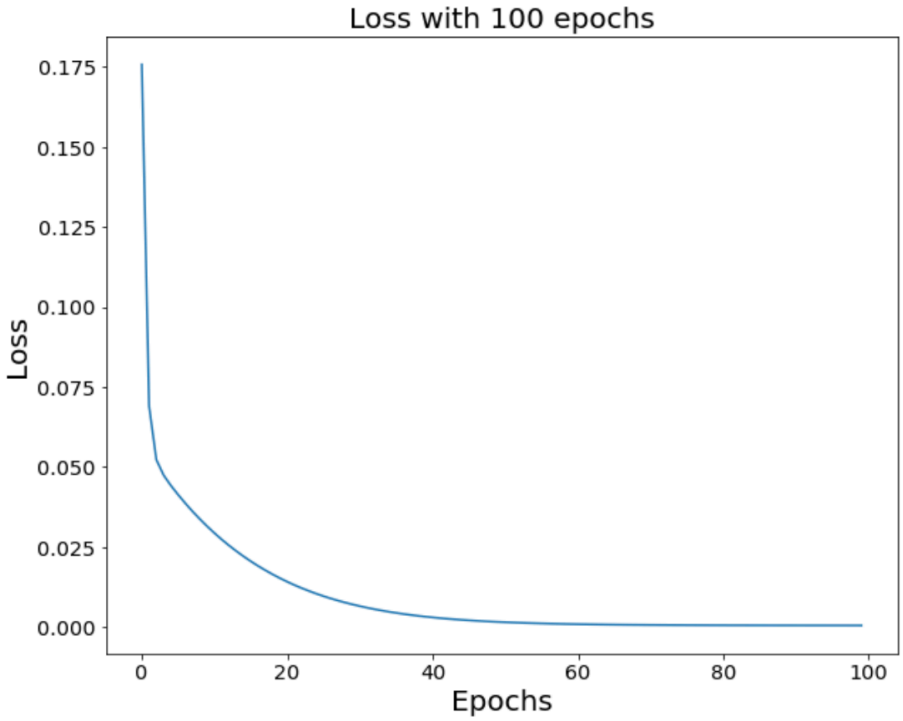

FIGURE 17. Loss of the model with 100 epochs

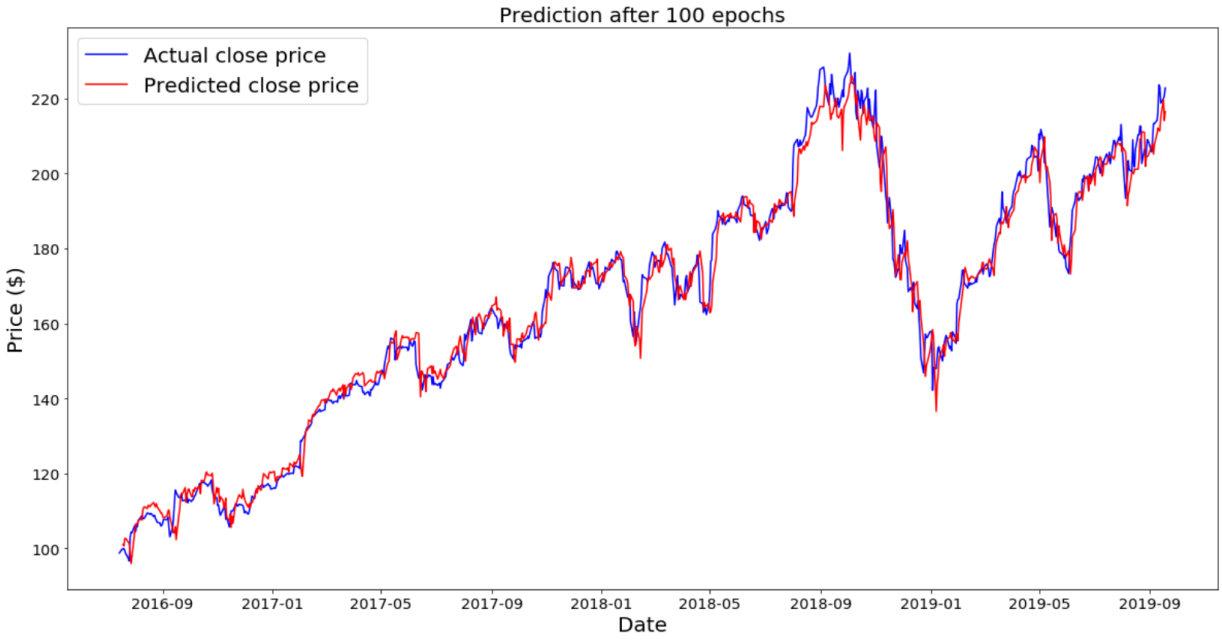

FIGURE 18. Prediction of the model with 100 epochs

From Figures 18 and 19 it can be seen how the predicted closing prices closely resemble the actual closing prices. The testing data, which is used for predicting, is new to the model – it has not seen that data before. It can be observed from Figure 19 showing where the training data stops and where the predictions start.

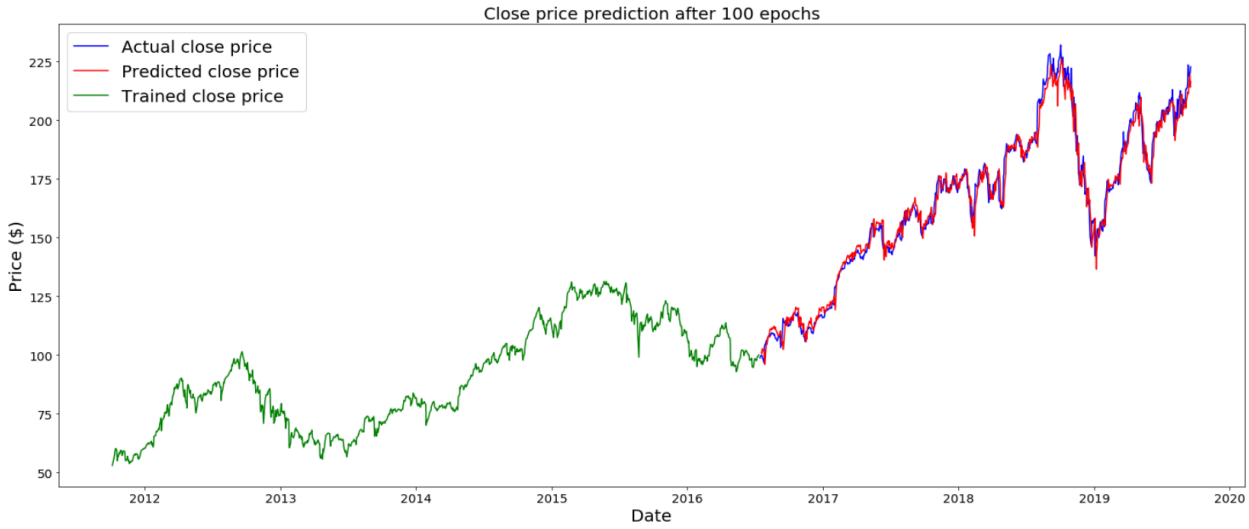

FIGURE 19. Training data, actual closing prices and predicted prices all plotted together

6 CONCLUSION

This thesis has shown different types of techniques, both fundamental and technical, to predict the stock market. While the EMH states that it is impossible to accurately predict the stock market (Chapter 2.1), advanced machine learning, such as LSTM neural networks, have opened a way to time-series forecasting not available before.

With the rise of many large internet companies, such as Google and Facebook, which store a lot of data from its users (Curran 2018), it is possible to obtain information about companies that would not otherwise be known. This could be achieved for example by following the location data that is sent from a high-ranking company executive's mobile phone, while in a meeting with a competitor, negotiating a buy. This information could then, theoretically, be added as a feature to a data set that is, in turn, trained into a predictive model. It is not an impossibility that through the Internet all information that can be available about a stock and a company will, in fact, be made available.

The predictive model shown in this thesis is a very simple implementation of an LSTM neural network. It is merely an example to demonstrate the capability and the potential of neural networks, especially in the context of time-series. The accuracy of a model is dependent on its data. The better the data set is, the more accurate the prediction will be. This thesis used 5 features in its data set, while real-world predictive models may have hundreds of thousands of features. These models will naturally be more accurate, but for the purpose of this thesis 5 features were more than enough.

The model presented in Chapter 5 was a success in predicting Apple Inc.'s stock market close prices. A more efficient model could be constructed with using more than just the previous day's information as the input for the neural network. More features could be added to the model, such as a sentiment analysis of the company and Google Trends data. For the scope of this thesis these features were not necessary.

Readers of this thesis must use the information explained and demonstrated about stock market prediction at their own discretion. This thesis does not encourage the use of the predictive model presented to make profit in the stock market. The predictive model is presented here purely for academic purposes and as an introduction to LSTM neural networks for anyone interested in them.

REFERENCES

Abdi H. & Valentin, D. & Edelman, B. 1999. Neural Networks. London: SAGE Publications Ltd.

Banerjee, S. 2018. An Introduction to Recurrent Neural Networks. Date of retrieval 16.01.2020 https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912

Bishop, C. 1996. Neural Networks for Pattern Recognition. New York: Oxford University Press Inc.

Boris, E. 2019. Demystifying Feature Scaling. Date of retrieval 23.01.2020 https://becominghuman.ai/demystifying-feature-scaling-baff53e9b3fd

Brixius, N. 2016. The Logit and Sigmoid functions. Date of retrieval 13.01.2020 https://nathanbrixius.wordpress.com/2016/06/04/functions-i-have-known-logit-and-sigmoid/

Brownlee, J. 2016a. Gradient Descent for Machine Learning. Date of retrieval 19.11.2019 https://machinelearningmastery.com/gradient-descent-for-machine-learning/

Bronwlee, J. 2016b. Supervised and Unsupervised Machine Learning Algorithms. Date of retrieval 19.01.2020 https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

Brownlee, J. 2019a. A Gentle Introduction to the Rectified Linear Unit (ReLU). Date of retrieval 15.01.2020 https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks

Brownlee, J. 2019b. How to Fix the Vanishing Gradient Problem with ReLU. Date of retrieval 16.01.2020 https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/

Callan, R. 1999. The Essence of Neural Networks. London: Prentice Hall Europe.

Chen, J. 2018. Shares. Date of retrieval 30.11.2019 https://www.investopedia.com/terms/s/shares.asp.

Chen, J. 2019. Stock Market. Date of retrieval 30.11.2019 https://www.investopedia.com/terms/s/stockmarket.asp.

Curran, D. 2018. Are You Ready? Here is All the Data Facebook and Google Have On You. Date of retrieval 14.01.2020 https://www.theguardian.com/commentisfree/2018/mar/28/all-the-data-facebook-google-has-on-you-privacy

De Marchi, L. & Mitchell, L. 2019. Hands-On Neural Networks: Learn How to Build and Train Your First Neural Network Model Using Python. Birmingham: Packt Publishing Ltd.

DeMuro, J. 2019. What is a Neural Network? Date of retrieval 20.11.2019 https://www.techradar.com/news/what-is-a-neural-network

Debarko, D. 2018. RNN or Recurrent Neural Networks for Noobs. Date of retrieval 16.01.2020 https://hackernoon.com/rnn-or-recurrent-neural-network-for-noobs-a9afbb00e860

Freeman, J. & Skapura, D.1992. Neural Networks: Algorithms, Applications and Programming Techniques. New York: Addison-Wesley Publishing Company, Inc.

Hayes, A. 2019. Technical Analysis. Date of retrieval 09.12.2019 https://www.investopedia.com/terms/t/technicalanalysis.asp

Haykin, S. 1994. Neural Networks: A Comprehensive Foundation. New Jersey: Prentice Hall Inc.

Heaton, J. 2015. Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks. Chesterfield: Heaton Research Inc.

Kang, E. 2017. Long Short-Term Memory (LSTM): Concept. Date of retrieval 18.01.2020 https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359

Kelleher, J. 2019. Deep Learning. Cambridge: The MIT Press.

Kostadinov, S. 2018. Recurrent Neural Networks with Python Quick Start Guide: Sequential learning and language modelling with TensorFlow. Birmingham: Packt Publishing Ltd.

Lucidarme, P. 2018. Simplest Perceptron Update Rules Demonstration. Date of retrieval 04.12.2019 https://www.lucidarme.me/simplest-perceptron-update-rules-demonstration/

Majaski, C. 2019. Fundamental vs. Technical Analysis: What's the Difference? Date of retrieval 09.01.2020 https://www.investopedia.com/ask/answers/difference-between-fundamental-and-technical-analysis/

Mankiw, N.G. & Taylor, P. 2014. Economics. Andover: Andrew Ashwin

MathWorks. 2019. Symbolic Hyperbolic Tangent Function. Date of retrieval 15.01.2020 https://se.mathworks.com/help/symbolic/tanh.html

Mishkin, F. 2018. Financial Markets and Institutions. London: Pearson Education Ltd.

Murphy, J. 1999. Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications. New York: New York Institute of Finance.

Nielsen, M. 2019. Using Neural Nets to Recognize Handwritten digits. Date of retrieval 19.11.2019 http://neuralnetworksanddeeplearning.com/chap1.html

Olah, C. 2015. Understanding LSTM Networks. Date of retrieval 18.01.2020 http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Pilbeam, K. 2010. Finance & Financial Markets. London: Palgrave Macmillan.

Pines, L. 2019. What Are Moving Averages in Technical Analysis. Date of retrieval 11.11.2019 https://commodity.com/technical-analysis/ma-simple/

Protasiewicz, J. 2018. Why is Python So Good For AI, Machine Learning and Deep Learning. Date of retrieval 16.01.2020 https://www.netguru.com/blog/why-is-python-so-good-for-ai-machine-learning-and-deep-learning

Ravichandiran, S. 2018. Hands-on Reinforcement Learning with Python: Master Reinforcement and Deep Reinforcement Learning Using OpenAI Gym and TensorFlow. Birmingham: Packt Publishing Ltd.

StockCharts. 2020. Date of retrieval 10.01.2020 https://school.stockcharts.com/doku.php?id=technical_indicators:moving_averages

Wikipedia. 2020. Artificial Neural Networks. Date of retrieval 14.01.2020 https://en.wikipedia.org/wiki/Artificial_neural_network